Learning Game 2.0: Support for Game Modding as a Learning Activity

Baptiste Monterrat, Elise Lavoué, Sébastien George

Learning Game 2.0: Support for Game Modding as a Learning Activity

Baptiste Monterrat[1,2], Élise Lavoué[1,3], Sébastien George[1,2]
[1]Université de Lyon, CNRS
[2] INSA-Lyon, LIRIS, UMR5205, F-69621, France
[3] Université Jean Moulin Lyon 3, MAGELLAN, LIRIS, UMR5205
baptiste.monterrat@insa-lyon.fr
elise.lavoue@univ-lyon3.fr
sebastien.george@insa-lyon.fr

Abstract: When we talk about video game, we observe that people who modified it are those who best knows its content. Thus we can consider game modifications as a way of knowledge appropriation. In this paper, we apply this model to learning games, positioning the research project in a Web 2.0 approach. If the content of a game can be learned by playing it, it can be more deeply understood by making this game evolving. The Web 2.0 is not defined by its technologies, but by the way of using it. However technologies 2.0 are developed to catalyse this participative way of use. Similarly with video games, specific tools and support are required. If they are simple enough to use, learning by game modding can be accessible to anybody without programming skills. In addition, game modding is a collaborative activity. Modders must be able to share their creations and to discuss their ideas. Thus, in our point of view, a game 2.0 environment is complete only if it allows users to play the game, to collaboratively modify it, and to share their creations. After an explanation of the game modding choice, we will present a model and a toolset for supporting such an educational activity.

Keywords: modding, game development kits, learning game 2.0, collaborative learning

1. Introduction

After writing for the "Reader's Corner" in the newspapers and calling the radio to give their opinion, the passive media consumers really became active contributors when using the Web 2.0 and its technologies. We believe video games are the new media taking part in this movement. During the last ten years, new forms of tools have emerged for budding developers who want to create games, like game factories or game modding tools. We can call them Game Development Kits (GDK).

Modding a game is the term used especially in computer game communities, to say “modifying the game” to create a new one. Some mods are a small game extension, and other ones are total conversions. Scacchi (2011) tried to summarize the uses of modding. He observed four types of game mods: "*user interface customization; game conversions; machinima; and hacking closed game systems*".

At the end of their tools study, Djaouti et al (2010) conclude that actual “*Gaming 2.0 offers some interesting ideas related to [...] game design process, but seems to lack awareness of the “Serious” potential of videogames.*". This paper aims to fill this gap by presenting a modding activity framework and tools. In the second part we will explain the reasons we chose game modding as a learning activity. In the third part we will present a model and tools supporting such an activity.

2. Modding: from player to learner

In this part, we will first present a state of the art about games co-creation, considering the player as part of the development process. Then we present the ways the user participation can be a learning activity.

2.1 The modding activity

2.1.1 Users’ potential

The most obvious way to mod a game is to modify its source code. The problem with this method is that it requires advanced skills in computer science. Fortunately, new tools have emerged, providing game players with new capabilities.

Tavares & Roque (2007) have argued the advantages for a game to be designed by a lot of players mixing a lot of ideas, instead of a few professionals. According to Volk (2008), "*the roles of game*

designer and game player is obviously not a binary one, since every level of participation can be found in the modding movement". In the same thought, we understand that video games need both professional design and fans ideas. As both roles exist, appropriate tools must be used. The most basic way for a player to customize his/her game is using the game setting. Most games allow the player to access parameters like the sound level, the display mode, or the level of difficulty. During the 80's, some games began to provide users with a level editor like *Lode Runner*[1], allowing them to change the initial state of the game (Djaouti, 2011). Nowadays, players finally have access to many game engines and have the possibility to make their own games by modifying others.

2.1.2 Technical support

Many kinds of tools have been created in order to simplify the programming. For instance, *Stencyl*[2] and *Flip*[3] are part of the most advanced ones. They allow the user to program the game without writing any code. Some blocs ("if", "then", "boolean", …) and existing functions can easily be drag and drop to build the game's behaviour. *Warcraft III editor*[4] proposes to do it with a more simple representation: triggers are rules composed of an event, some conditions and some actions. When an event occurs (like a character entering in an area, or a building being attacked), the conditions are checked and the actions are triggered. At last, *Kodu Game Lab*[4] and *Game Develop*[5] are using one of the simplest way we found to define a game: rules composed by conditions and actions only. The game engine acts as if the conditions are permanently checked and if they are verified, the corresponding actions are triggered. This system is simple enough for children to make games.

Just like the web 2.0 grew up thanks to the WYSIWYG editors, the game 2.0 is becoming more accessible to everyone thanks to new game editors generation that let people see their changes within the game. More and more tools offer a visual editor for the game scenes. Their initial state can be modified by dragging and dropping objects directly from a library to the game. *Kodu* also allows an access to the game engine directly from the game: selecting a character leads to the rules defining its behaviour. For more details on this part, Djaouti *et al.* (2010) presented a study of fifteen "gaming 2.0" tools, and the way they can help the design of serious games.

2.1.3 Using independent tools

Volk (2007) have been one of the first to talk about "*game development 2.0*". He observed that consumers do not freely participate, "*they are monitored by the producer in terms of market research*". Many other authors warn about this phenomenon. Sotamaa (2005) describes the game modding competitions as a way of managing free labour made by the modders. Affordances in the development tools have also been highlighted by Scacchi (2010). These "*socio-technical affordances serve to organize and govern the actions of the people who develop and share their game mods […] and profits for the game development studio, publisher, and retailer.*" At last, Postigo (2008) have shown the limits of modding copyrighted games, when modders are not authorized to share their creations. All these observations leads to a preference for using free software, which is a good way to stay focused on educational interests.

2.1.4 Community support

Loh and Byun (2009) have created a serious game by modding *NeverWinter Nights 2*. They relate their experience as “game developers postmortem” which are developers who have completed their game, and “*are documenting what went right and what went wrong along the way*”, so others “*can repeat the successful parts of the development process, and avoid the pitfalls [...] encountered along the way.*” These experienced developers and beginners organize themselves into communities built around a modding software. They share ideas about the ways to mod through collaborative tools (e.g chat, forums, in-game conversations). Strong support exists within these communities, allowing new developers to quickly overcome the problems they meet.

2.2 Learning by modding

2.2.1 Learning computer science

In various studies, students have been allowed to create mods by themselves. McAtamney (2005) made them using *Crytek* engine to design the site of the new campus of their university. Through this experience, they learnt how to use C++, direct X and the open source scripting language LUA, while improving their skills in maths, physics, 3D design and game events. In a funnier activity, El-Nasr & Smith (2006) shown that existing modding tools are adapted to different types of learners. They first

used *Warcraft III Editor* with high school students to make them create new games in only three days, in order to learn the basics of algorithmic. Then they used *Web Driver* and *UnrealEngine 2.5* with computer science students in a course with higher educational goals. In addition to technical skills acquired, all the students started becoming familiar with the software development process (designing, coding, testing), because it is quite the same as a mod development process (Cignoni, 2001).

2.2.2 Learning the content

Through all these experiences, modding have been used to teach computer science or mathematical knowledge. In this paper we propose to focus on a particular activity, which is relatively few studied: modding to learn the content of the learning game. For example if the game is about mathematics, the modder will learn mathematics. Oblinger (2006) explains that a learning game efficiency depends on the level of involvement of the player. Our aim is to improve this involvement by allowing the user to modify the game. A modifiable game provides players with a new way to practice game based learning. This method is part of the constructivist approach, providing a way of learning by doing (Lave, 1991).

Moshirnia (2007) has studied modding as a way for teachers to make learning games. They taught the American Revolution with a mod of the game *Civilisation IV*. The learning game design is part of the teacher's work. The teacher is the one who knows how to teach, whether it is through a game or another way. As some students already knew the game, they were really interested and deeply involved in the learning activity. With game modding, we now propose to the students to play the role of the teacher. Making a learning game is a way of teaching. Also, in order to make a fully coherent course or game, a teacher has to learn new things. Moreover a teacher reinforces his knowledge while teaching, because transmitting requires a deeper level of understanding. Accordingly, learners as modders can strengthen their knowledge like teachers do.

2.2.3 Still being a player

We want this learning activity to be accessible to anybody, whether they have programming skills or not. We think that co-design from scratch is too difficult and provides the user with multiple useless options. With game modding, the game structure already exists and the learners can work exclusively about the game content. For example, imagine a game designed for language learning, where the players have to move the character to an object when they hear the name of the object. If the players have to create the all game, they will waste time working on the character creation and the way to move it. With game modding, we assume that the teacher already created the game and the character, and then learners can focus on adding new sounds and new objects to the game, which is directly related to their learning goals.

Also with game modding, learners may look for new information by themselves, until they may learn things beyond the teacher's knowledge. According to the Magic Bullet model (Becker, 2011), learners will improve the part of external learning, and include by themselves this new knowledge within the game. Thus the next players will have to learn it too to get through the game. This way of developing also makes the knowledge embedded in the game being evolutive, in order to always be up to date, like a wiki, in accordance with the constantly evolving world. Such a game can also be well adapted to different kinds of learners by being modified by the learners themselves.

2.2.4 Player's motivations

Sotamaa (2008) presents different motivations for modders to mod: playing with the game editor, researching, co-operating and making artistic expression. We also see the pleasure of making something immediately re-usable. Generally when students learn during a class, their homework, projects and course notes are rarely reused. We propose learners to mod in small groups, and then to share their projects with the class. Since they know that their creation will be played at least by their classmates, or by the next year students, we think they want to make a quality work.

Also, many aesthetic elements are generally ignored when designing serious games, because these elements require a lot of time and are not the main objective. As an example, a changing weather is not necessary if the game goal is to meet avatars and talk with them, and the sky can be ignored. However, even if your game scenario is the best one, nobody will be interested in playing if the character is a blue point moving on a black background. Details often are a reason that makes the

player liking the game or not. An advantage with game modding is that the all environment already exists. The modder only has to focus on particular points (for instance defining the objects place and the interactions between them).

2.2.5 Learning through collaboration
According to Scacchi (2011),"*Modding is also a practice for learning how to work with others*". A modding person learns how to work in team and manage group projects. During the experiment of El-Nasr and Smith (2006), the students first learnt to divide the tasks among groups of two and share their skills. They then went beyond, exchanging with other groups. They understood by themselves that communicating about their project and discovering others would be beneficial. In bigger projects, modding also teaches how to manage a team and sometimes how to resolve conflicts. For example, Loh and Byun (2009) understood that when they develop the mod, they cannot freely change the planned format of the game without offending the writer of the team.

As well, being a bigger team is beneficial for the modding project in return. When the number of persons involved in the game modding increases, there are more risks to make a mistake, but there are also more people able to detect the errors and to correct them. Ang et al. (2005) have studied this "*self-regulating mechanism*" in wiki communities. The same phenomenon can occur around learning games, as they are also collaborative media.

3. Proposed framework for supporting learning by modding
Through the previous part, we suggested a new kind of educational activity based on game modding. In this part we aim to expose framework and tools for making modding being an educational activity.

3.1 Model and tools for a modding activity

3.1.1 The game and the GDK
The GDK is the tools package coming with a game. On the one hand, it has to be simple enough for a quick start, without programming skills needed. It's necessary for modders to learn how to mod, but we want this step to be as short as possible, so they can quickly begin to learn about the game's content by modifying it. On the other hand the GDK has to be rich and powerful for allowing deep structure modifications to the game and define evolved behaviours. A system of rules composed of conditions and actions seems to be a good compromise. We also observed that video games are generally composed of scenes, which can be interpreted like “levels”, “maps” or “worlds” depending on the game. At last, we will consider here all the game elements (textures, sounds, characters, …) as objects. In figure 1 we present a model that underpins our work.

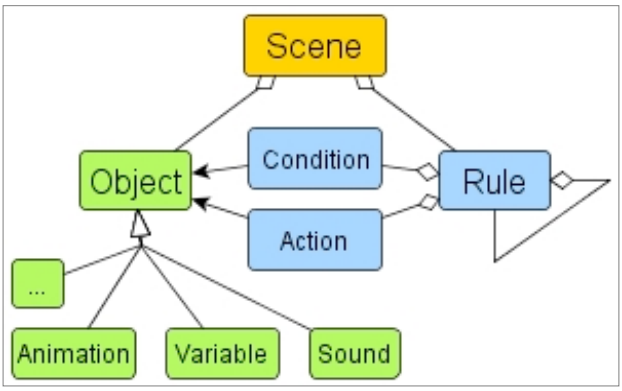

Figure 1: The model underpinning the game and the GDK

The conditions and actions are related to each kind of object. We give examples below.

- conditions: “Is the sound <A> being played at this time?”, “Is the key <Left> of the keyboard is pressed at this time?”.
- actions: “Play the sound <A>”, “Move the image <B> 10 pixels to the left”.

If a rule has several conditions, they need to be all verified to trigger the actions, like if they were separated by a logical AND. The way of simulating a OR is to put the others conditions in other rules.

In this way, a budding developer can design advanced conditions without taking boolean logic courses. As all logical formulas can be converted into disjunctive normal form, this system has no limit in conditions expressiveness. Also some tools (e.g. *Kodu* and *Game Develop*) allow the creation of sub-rules in order to avoid the rewrite of conditions. A sub-rule will be checked only if its super-rule have been triggered. A set of sub-rules is a way of representing the braces after a condition in the language C. Thus it is also quite easy for the modder to understand how a set of rules will be interpreted by the game engine, while the model is also very expressive, accordingly to the amount of functions provided with the objects types.

The users need to build the game at the end of the development process, but they also need to test it many times while modifying the objects and rules. In particular non professional developers need to see the effectiveness of their changes into the game. That's why the compilation process is a problem. Firstly repeated compilations may take a long time. Secondly, within a long game, the developers will have to start the game from the beginning and get through it any time they want to test the final situation. Accordingly we have to allow quick switches between the modding interface and the game test interface, by using a game engine interpreting the game rules in real time, or compiling the scenes independently. This approach is allowed by the game division in scenes.

3.1.2 The social platform

As explained through the previous part, game modding becomes interesting and efficient when done collaboratively. This is especially true when the activity is educational. On the one hand, modders need to be able to share ideas about the game they are imagining together (e.g. they have to discuss the place of an image, the level of difficulty or the specific behaviour of the game). On the other hand, they need to share game elements (e.g giving a set of rules for another modder to be included within her/his game, or downloading a scene to test it). This implies the use of a social platform meanwhile they mod, in order to support conversations and the share of elements (games, scenes, objects and rules).

However, switching from the game to the GDK, and from the GDK to the social platform could be a barrier for learning. We argue that the game and its corresponding GDK have to be integrated with the social platform. That's why these 3 facets of game modding have to be parts of a unique tool. According to McAtamney et al (2005) "*it is also unusual to have all the level editing requirements [...] packaged into one program*.". Furthermore that matches to the definition of Game 2.0 given by Djaouti in his thesis (2011): "*any application allowing a user to create, share and play to a game content*".

3.2 Model and tools for a learning by modding activity

The model depicted earlier allows us to describe the elements of a complete game 2.0. We now have to focus on making all of this an educational activity, in order to build the concept of learning game 2.0. For this purpose we must take into account particularities of educational area, like the pursuit of educational goals, the presence of a teacher, and the possibility to evaluate achievements. The teacher must be the "driver" of this activity, because s/he is the one experienced for taking in account the level of the students, their age, the available time, etc... Her/his tasks are:

- to design all the course scenario (which includes modding), according to the educational goals.
- to create the starting game which will be modified by the learners.
- to monitor the activity, to help the students break the deadlocks, and make sure they follow the objectives.
- to evaluate the learning, and possibly the learners.

We extend the previous model in order to take into account educational aspects, as presented on Figure 2.

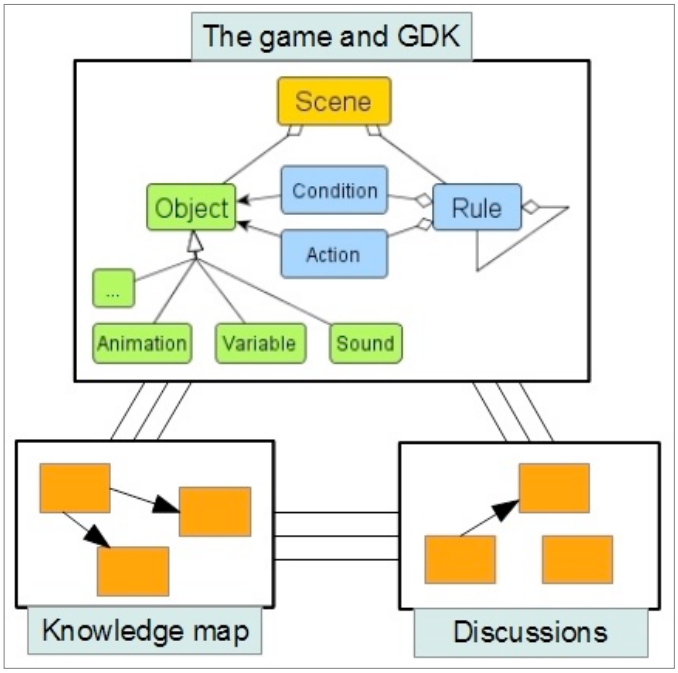

Figure 2: The model of a learning by modding activity.

The things that students have to learn are represented on a knowledge map, and the elements of this map are related to the game elements. As examples:

- "Learning the law of gravity" would be related to the game rules simulating the law in the game.
- "Learning to recognize animals" would be related to the zoo scene.

This is a way for the teacher to know what the students are learning when they are working on a given scene. Regarding ideas sharing, it seems important to connect each discussion to the related elements in the GDK, and also to the knowledge map. These "contextual discussions" have proven to be useful in educational situations (George, 2004). This is also a way for a distant teacher to guide the students' work and answer their questions.

3.3 Experimenting on an example

We plan to experiment the validity of our model and tools. Students will have to make evolving scenes of a learning game in Esperanto, to learn this language.

3.3.1 The game

For our experiment, we choose to teach the basis of the language Esperanto[6]. The game owns a central scene where the player's character will find doors to access others. These scenes will be the places to learn things about Esperanto and to earn points. Later, points will be necessary to access to other scenes with a more difficult level. This is a way to evaluate the language skills of the player through the game. For a better understanding, we give below an example of scenes:

- In the scene 1, the players can read the name of an object in Esperanto. Then they have to find the corresponding object to earn points in a given time, avoiding the other objects. In this scene they can learn vocabulary.
- In the scene 2, the players will hear a word in Esperanto. Then they have to find the letters heard, and to move them into the correct order to recompose the word. In this scene they can learn the pronunciation.

3.3.2 The learning scenario for the experiment

Below we present the scenario of the global learning activity for the experiment, which includes the game modding as one of the steps.

1. At first the students will take a one hour Esperanto course to learn the basis of the language.
2. Then they will play *the game* and make comments about its lakes or defects.

3. After that, they will mod the game during one hour, with a close help of the teacher. This step aims to make the students familiar with the modding tool. Steps 1 to 3 will take place in a classroom with a teacher.
4. Students then will be divided into two teams of four and go back home. They will have one week to collaboratively mod the game remotely. Each team will also be divided for not working more than two on the same scene together. They will be asked to work at least two hours on the project during this week. For example, they may add objects words to scene 1, add sounds to scene 2, or raise the level of difficulty. At the end of this step, a questionnaire will let us know if the students learnt things while modding.
5. At last, the members of team 1 will play the mod created by team 2, and the team 2 will play the mod created by team 1. At the end of this step, a questionnaire will let us know if the mods created are fun and if they allow an efficient learning.

In order to evaluate the provided model and tools, the five steps will be repeated with two groups of eight students. The first group will be provided with the game and the modding tool only, the second group with the whole platform we propose. The feedback will let us know the more adapted conditions and the impact of the proposition.

3.3.3 Our learning game 2.0 system

As described, this activity requires a platform supporting game modding, playing, sharing, and discussions. This platform also has to be simple enough to be quickly understood, and rich enough to allow users to represent evolved behaviours. Such a program doesn't exist yet. We decided to combine various softwares in order to answer the requirements. We chose *Game Develop* as the heart of this prototype. *Game Develop* is a software allowing beginners as well as experienced game developers to create any kind of 2D games. Its games rules system matches the model we described above. Also the software proposes a quick scene compilation for rapid testing, and a full game compilation to make a game file.exe easily shareable. Thus *Game Develop* meets the specifications of accessibility to novices and expressiveness.

In parallel, we manage the project repertory with SVN, which allows several users to work remotely on the same project, keeping the different files up to date. Therefore, we allow the sharing of game objects. For the experimentation, we chose to make available one discussion related to each scene, as one discussion per object or rule would have been too much according to the low number of participants. We made the discussions system with the languages html and php. For the ease of use, a small program makes the link between *Game Develop* and the discussions application. When the user switches to work on another scene, the program's role is to display the corresponding discussion. In that way the modder can immediately read the co-modders ideas about this scene.

4. In conclusion

In conclusion, in this article we have explored several works about game modding as a way to learn. We also advocate that game modding could be interesting for any kind of learning and not only to learn programming. Then we have shown that this activity can be accessible to anybody if appropriate tools are provided, and that learning by modding games becomes really interesting in collaboration. We finally propose a model and tool supporting this activity, and an example of use of such an activity.

This work aims to open a door to new pedagogical activities, providing the learners with more powerful intellectual and technical tools, allowing them to express their potential. The students can be the new leaders of their learning, enabled to progress at their own pace and level.

References

Ang, C. S., Zaphiris, P., and Wilson, S. (2005) Wiki-supported Collaborative Narrative Construction in Game Communities, *The ECSCW'05 workshop on "Computer Games CSCW"*, Paris.

Becker, K. (2011) "The Magic Bullet: A Tool for Assessing and Evaluating Learning Potential in Games", *International Journal of Game-Based Learning, Vol 1, No. 1, pp 19–31.*

Cignoni, G. (2001) Reporting about the Mod software process. *Software Process Technology*, pp 242–245.

Djaouti, D., Alvarez, J., and Jessel, J. P. (2010) Can Gaming 2.0 help design Serious Games?: a comparative study. *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games,* pp 11–18.

Djaouti, D. (2011) *Serious Game Design - Considérations théoriques et techniques sur la création de jeux vidéo à vocation utilitaire*, computer science thesis, université de Toulouse III - Paul sabatier.
El-Nasr, M. S., and Smith, B. K. (2006) Learning through game modding. *Computers in Entertainment (CIE)*, Vol *4, No.* 1, Article 3B.
George S. (2004) Contextualizing discussions in distance learning systems, *Proceedings of the 4th IEEE International Conference on Advanced Learning Technologies (ICALT 2004)*, Joensuu, Finland, pp 226–230.
Lave J., Wenger E. (1991) *Situated learning: Legitimate peripheral participation*, Cambridge University Press, New York.
Loh, C. S., and Byun, J. H. (2009) Modding Neverwinter Nights into serious games, *Digital Simulations for Improving Education: Learning Through Artificial Teaching Environments*, pp 408–426.
McAtamney, H., O'Shea, B., and Mtenzi, F. (2005) Using the Crytek game engine in the Dublin Institute of Technology, *Proceedings of the 7th International Conference on Computer Games*, Angoulême, France.
Moshirnia, A. (2007) The educational potential of modified video games. *Issues in informing science and information technology*, Vol *4*, pp 511–521.
Oblinger, G. (2006) *Games and Learning, Educase quarterly, Vol 29, No. 3, pp 1–7.*
Postigo, H. (2008) Video Game Appropriation through Modifications: Attitudes Concerning Intellectual Property among Modders and Fans. *Convergence: The International Journal of Research into New Media Technologies*, Vol *14, No.* 1, pp 59-74.
Scacchi, W. (2010) Computer game mods, modders, modding, and the mod scene. *First Monday*, vol *15, No.* 5.
Scacchi, W. (2011) Modding as a Basis for Developing Game Systems. *Proceedings of the 1st international workshop on Games and software engineering*, Waikiki, Honolulu, HI, USA, pp 5–8.
Sotamaa, O. (2005) Critical Perspectives On Computer Game Mod Competitions. *Proceedings of DiGRA 2005 Conference: Changing Views–Worlds in Play,* Vol 16.
Sotamaa, O. (2008) When The Game is Not Enough: Motivations and Practices among Computer Game Modding Culture. *Games and Culture,* Vol 5, No. 3, pp 239–255.
Tavares, J. P., and Roque, L. (2007) Games 2.0: Participatory Game Creation. *Proceedings of the 6th Symposium on Computer Games and Digital Entertainment. São Leopoldo, Brazil*.
Volk, D. (2007) Game development 2.0. *Proceedings of the 2007 conference on Future Play, Toronto, ON, Canada,* pp 225–228.
Volk, D. (2008) Co-creative game development in a participatory Metaverse. *Proceedings of the Tenth Anniversary Conference on Participatory Design 2008, Bloomington, IN, USA,* pp 262–265.

[1] Lode Runner (1983) was one of the first games to include a level editor.
[2] Stencyl http://www.stencyl.com/
[3] Flip http://www.flipproject.org.uk/
[4] Kodu http://research.microsoft.com/en-us/projects/kodu/
[5] Game Develop http://www.compilgames.net/index.php?lang=en
[6] Esperanto is an international language created by Dr Zamenhof (1887).